Pending Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for operating a code cache in a dynamic instruction translator comprising the steps of:

storing a plurality of instruction translations in a cold partition in a code cache memory that only stores code instructions generated by a dynamic instruction translator;

determining whether an instruction translation that has been stored in the cold partition is hot; and

moving the instruction translation to a hot partition in the code cache memory when an instruction translation has been determined to be hot.

2. (Currently Amended) A method as defined in claim 1, wherein the step of determining whether an instruction translation is hot comprises:

maintaining a different associated counter for each of a plurality of instruction translations in the cold partition of the code cache memory;

incrementing or decrementing the count in the associated counter each time its associated instruction translation is executed; and

concluding the determination that an instruction translation is hot if the count in the associated counter reaches a first threshold value.

3. (Currently Amended) A method as defined in claim 1, wherein said hot partition is contiguous and disjoint from said cold partition in said code cache memory.

4. (Currently Amended) A method as defined in claim 2, wherein said maintaining an associated counter step comprises maintaining counters in a data structure external to said code cache memory.

5. (Currently Amended) A method as defined in claim 4, further comprising the step of at least temporarily delinking blocks of instruction translations stored in said cold partition so that control exits the code cache memory in order to perform the incrementing or decrementing step.

6. (Currently Amended) A method as defined in claim 2, wherein said maintaining within said code cache memory an associated counter step comprises maintaining one of said associated counters for each entry point into a plurality of the instruction translations in said cold partition of the code cache memory.

7. (Previously Presented) A method as defined in claim 2, wherein said maintaining an associated counter step comprises logically embedding update code on an arc between two instruction translations.

8. (Original) A method as defined in claim 2, wherein said maintaining an associated counter step comprises maintaining one of said associated counters for each machine cache line in an associated microprocessor.

9. (Previously Presented) A method as defined in claim 2, wherein said instruction translation moving step comprises sampling a plurality of said associated counters on an intermittent basis to determine if the count therein has reached said threshold value.

10. (Currently Amended) A method as defined in claim 1, further comprising the steps of:

determining if a number of hot instruction translations in said hot partition of said code cache memory exceeds a second threshold value; and

if said number of said hot instruction translations exceeds said second threshold value, then expanding the size of said hot partition in said code cache memory by adding thereto an expansion area contiguous to said hot partition.

11. (Previously Presented) A method as defined in claim 10, further comprising the step of

removing all cold instruction translations from said expansion area and storing said removed instruction translations in said cold partition.

12. (Currently Amended) A method as defined in claim 2, wherein the maintaining an associated counter step comprises maintaining an associated counter for all instruction translations in the cold partition of the code cache memory.

13. (Currently Amended) A system for a code cache in a dynamic instruction translator comprising:

a code cache memory that only stores code instructions generated by a dynamic instruction translator;

a cold partition and a hot partition in said cache memory;

logic for determining whether an instruction translation that has been stored in the cold partition is hot; and

logic for moving the instruction translation to a hot partition in the code cache memory when an instruction translation has been determined to be hot.

14. (Previously Presented) A system as defined in claim 13, wherein the logic for determining whether an instruction translation is hot comprises:

logic for associating a different counter for each of a plurality of instruction translations stored in the cold partition of the cache memory;

logic for incrementing or decrementing the count in the associated counter each time its associated instruction translation is executed; and

logic determining if the count in the associated counter reaches a first threshold value.

15. (Currently Amended) A system as defined in claim 13, wherein said hot partition is contiguous and disjoint from said cold partition in said code cache memory.

16. (Currently Amended) A system as defined in claim 14, wherein said counters are maintained in a data structure external to said code cache memory.

17. (Currently Amended) A system as defined in claim 16, wherein said incrementing or decrementing logic further comprises logic for at least temporarily delinking blocks of instruction translations stored in said cold partition so that control exits the code cache memory in order to perform the incrementing or decrementing of the count.

18. (Currently Amended) A system as defined in claim 14, wherein said logic for associating counters comprises logic for maintaining one of said associated counters for each entry point into a plurality of the instruction translations in said cold partition of the code cache memory.

19. (Previously Presented) A system as defined in claim 14, wherein said logic for moving the instruction translation comprises logic for sampling a plurality of said associated counters on an intermittent basis to determine if the count therein has reached said threshold value.

20. (Currently Amended) A system as defined in claim 13, further comprising:

logic for determining if a number of hot instruction translations in said hot partition of said code cache memory exceeds a second threshold value; and

if said number of said hot instruction translations exceeds said second threshold value, logic for expanding the size of said hot partition in said code cache memory by adding thereto an expansion area contiguous to said hot partition.

21. (Previously Presented) A system as defined in claim 20, further comprising:

logic for removing all cold instruction translations from said expansion area and storing said removed instruction translations in said cold partition.

22. (Currently Amended) A system as defined in claim 14, wherein the logic for associating a counter step comprises logic for maintaining an associated counter for all instruction translations in the cold partition of the code cache memory.

23. (Currently Amended) A program product, comprising a computer usable medium having computer readable program code embodied therein that, when executed, directs a computer to manage a code cache memory in a dynamic instruction translator, the program code comprising:

code for storing a plurality of instruction translations in a cold partition in a code cache memory that only stores code instructions generated by a dynamic instruction translator;

code for determining whether an instruction translation that has been stored in the cold partition is hot; and

code for moving the instruction translation to a hot partition in the code cache memory when an instruction translation has been determined to be hot.